Notes on applying segmentations and relevant geometry.

DAC. 2013/12/10.

Background and Assumptions.

1. Segmentations may theoretically apply to 2D (no spatial information) or 3D (with spatial information) images, and may themselves be 2D or 3D.
2. 3D segmentations and the 3D images to which they apply, may or may not be parallel and/or regularly sampled ... for argument's sake, a 3D dataset that for which all slices are parallel and regularly sampled is referred to herein as a "regular 3D volume".
3. 2D segmentations (no spatial information) have explicit references to image/frame to which they apply (regardless of whether those images/frames themselves are 2D, 3D or a regular 3D volume); the explicit reference is encoded in the Derivation Image functional group (PS 3.3 C.7.6.16.2.6) within either the Shared Functional Group Sequence item or the frame-specific item of the Per-Frame Functional Groups Sequence; the reference consists of Derivation Image Sequence (0008,9124) within which is nested Source Image Sequence (0008,2112), which contains Referenced SOP Instance UID (0008,1155) +/- Referenced Frame Number (0008,1160).
4. The Study and Series containing any referenced image/frame are listed in the Common Instance Reference Module (C.12.2), which consists of the Referenced Series Sequence (0008,1115) used for references to series in the same Study as the Segmentation object, and the Studies Containing Other Referenced Instances Sequence (0008,1200) used for references to series in different Studies.
5. 3D segmentations may theoretically be rendered on any images/frames that overlap in their 3D geometry; this means that there may be multiple series (or multi-frame images) in a Study (or even multiple Studies that share the same Frame of Reference UID) that are candidates for rendering; indeed even images that are not sampled in-plane or cross-plane with the same spacing, or are orientated differently (such as orthogonal MPRs) may be appropriate candidates on which to render the segmentation (i.e., a 3D segmentation is an "object" in space and may be rendered and viewed arbitrarily in space)
6. If there is one or more series (or multi-frame images) that matches in orientation and sampling rate (as opposed to other series that don't), then those series are arguably more plausible candidates for rendering the segmentation than others (if the segmentation is not going to be rendered on all series).
7. 3D segmentations may or may not have explicit references (encoded in the same manner as 2D segmentations); if explicit references are present, the

meaning defined in the standard is that the segmentation was derived from the referenced images/frames (PS 3.3 A.51.5.1; clarified in CP 1265); this does not mean that the 3D segmentation cannot be rendered on other images/frames that share the same 3D geometry even if not referenced, theoretically; but it may be expedient to render them only on the referenced series.

8. A single segmentation object may reference frames of different multi-frame images, not just frames in a single multi-frame image, or different single frame images; this might be the case, for example, if Concatenations were used to split up a large multi-frame volume into smaller multi-frame images.
9. For 3D segmentations and 3D images, in the general case, the problem can be considered as one of resampling, such as might be used for image fusion of registered multi-modality datasets; i.e., a general solution would be to resample the 3D segmentation data (whether regularly sample or parallel slices or not) into the same 3D orientation and sampling rate as the "target" image data set, so as to create a 1:1 voxel correspondence between the voxels of each segment in the segmentation and the image voxels. E.g., if one were using a 3D rendering package like VTK, one might use the vtkImageReslice class to perform this ("http://www.vtk.org/doc/release/5.0/html/a01577.html"). A complicating factor is that segmentations are often/usually binary (rather than continuous probability or fractional occupancy), so interpolation needs to account for that (e.g., with a threshold at edge values that are not unambiguously 0 or 1).
10. In the absence of 3D resampling support, common degenerate cases in which the slices are parallel and overlapping may be handled explicitly by treating frames individually, finding the "closest" (sufficiently close) frame using 3D location information, ignoring cross-plane sampling, and if necessary, resampling (interpolating) only within the plane of matching slices (i.e., accounting for different Pixel Spacing values).
11. Each segment in a segmentation object is essentially a separable problem from all other segments in that object, and the following discussing treats each segment independently, as if it were its own distinct set of 2D or 3D sets of slices.
12. Matters of timing information in images are not considered. In the absence of explicitly referenced images, the segmentation objects do not have any explicit means of distinguishing which subset of multiple potential target images in the same location but obtained at different times are intended for rendering. E.g., a segmentation may be obtained from a portal venous phase liver image, but there is no way (other than an explicit image reference) of knowing that, or preventing rendering on other phase (pre-contrast, arterial, delayed) images at the same location but different times. It might be conceivable to use the Dimension Organization UIDs and indices to address this concern, but since these are not yet widely used in image objects in the first place, this is an area for future investigation. In the absence of explicit image references, a segment should be rendered on all target slices with matching locations (perhaps at the user's discretion).

Use Cases.

Single or multi-frame 2D Segmentation and single or multi-frame 2D or 3D referenced image

1. A referenced image is present for each frame of the segmentation and each segmentation frame can be processed independently
2. Frame of Reference UID, Image Position (Patient) and Image Orientation (Patient) will not be present in the segmentation; they may be present in the referenced image (3D) but may be ignored; whether a 3D referenced image is a regular 3D volume or not is irrelevant.
3. Per PS 3.3 A.51.5.1, the size and spacing of segmentation Pixel Data and the referenced image Pixel Data are required to be identical (since there is no other way to match corresponding voxels).
4. The pixels of (each or the only frame of the) segmentation correspond 1:1 with the pixels of the (only or referenced frame of the) referenced image.
5. If the referenced image is multi-frame (Number of Frames > 1) and there is no Referenced Frame Number in the reference, the frame cannot be determined (error).

Single or multi-frame 3D Segmentation and single or multi-frame 3D images with the same Frame of Reference UID, no explicit references, and identical geometry

1. A set of (loaded) images with the same Frame of Reference UID is identified (regardless of Study, though for simplicity, could constrain to be the same Study Instance UID too).
2. Frame of Reference UID, Image Position (Patient) and Image Orientation (Patient) are present in both the segmentation object and the images (nested in the appropriate sequences depending on the SOP Class).
3. For each segmentation frame, there is one or more image frames with identical values of Image Orientation (Patient) (i.e., parallel), Image Position (Patient) (i.e., same TLHC), same Pixel Spacing (i.e., same in-plane sampling), same Rows and Columns (i.e., same size) (within floating point tolerance of <0.01mm for distances, and < 0.001 for unit vector components, for example).
4. For such matching frames, the pixels of the segmentation frame correspond 1:1 with the pixels of the image frame.

Single or multi-frame 3D Segmentation and single or multi-frame 3D images with the same Frame of Reference UID, no explicit references, and parallel matching slices but otherwise different geometry

1. A set of (loaded) images with the same Frame of Reference UID is identified (regardless of Study, though for simplicity, could constrain to be the same Study Instance UID too).
2. Frame of Reference UID, Image Position (Patient) and Image Orientation (Patient) are present in both the segmentation object and the images (nested in the appropriate sequences depending on the SOP Class)

3. For each segmentation frame, find image slices that are parallel, i.e., Image Orientation (Patient) values are equal (within floating point tolerance, say perhaps absolute difference of components is <=0.01)
4. Amongst the parallel slices find the "closest" slice (or set of closest slices, if more than one is equally close); "closest" being defined to be that with the shortest distance along the normal vector to the plane orientation (perpendicular to the plane); the normal vector is the vector cross-product of the plane row and column vectors; the distance from a parallel plane passing through the 3D coordinate system origin (0,0,0) along the normal can be computed for each candidate slice as the vector dot product of the normal vector and a vector from the origin to the TLHC of each slice (i.e., the Image Position (Patient) X, Y and Z values treated as a vector); the "closest" (slices) are then those whose distance along the normal are closest to the distance along the normal of the segmentation frame.
5. Given one or more "closest" slices, are they "close enough" to consider rendering on (i.e., one is expecting a near identical location, considering floating point calculation tolerance); an absolute difference of the distance along the normal (dot product) of say <=0.01mm accounts for floating point tolerance.
6. For each matching slice, the offset (different TLHC) needs to be accounted for. Given the 3D TLHC of the segmentation frame (its value of Image Position (Patient)), one needs to compute the image-relative 2D location (offset in terms of row and column on the image frame). If the image (series) is a regular 3D volume then one can perform a volume-based calculation, but a simpler calculation is possible given the slice already selected (which does not depend on a regular 3D volume).
7. Different sampling (different Pixel Spacing) needs to be accounted for when finding "in plane" pixels from the segmentation frame to apply to the image frame. The scaling factor to use when interpolating in-plane is the ratio of segmentation spacing to the image spacing, for the corresponding spacing between adjacent rows and spacing between adjacent columns (given that pixels may theoretically be non-square, and that the aspect ratio of the segmentation and the image may be different). A simplification is only to support square pixels in both the segmentation and the image, allowing a single scale factor for both row and column directions. A further simplification is only to support the same spacing in both the segmentation and the image (i.e., allow offset and different area only but not different spacing), requiring no scaling.

Single or multi-frame 3D Segmentation and single or multi-frame 3D images with the same Frame of Reference UID, no explicit references, and parallel slices that do not exactly match and may have otherwise different geometry

1. This case is the same as exactly matching slices, except that the tolerance required for the "closest" slice is relaxed. A priori, it may not be possible to distinguish this use case from that of exactly matching slices until one fails to find "closest" slices with a tolerance of close to zero.

2. Given one or more "closest" slices, are they "close enough" to consider rendering on? One wants to exclude slices that are too far away, but permit slices that are pretty close, even if not in an identical location; the choice of a tolerance value is a judgment call and may require some tuning, and may depend on how thinly the slices are spaced; a difference of <=0.1mm (for thin slices) or <=1mm (for thicker slices) might be appropriate. The risks in this approach are that a segment may be rendered on the "wrong" slice (if too large a tolerance value is selected), or not rendered at all (if too small a tolerance value is selected). In the latter case, it may be appropriate to warn the user that segments are present that could not be rendered.
3. Arguably, given the use cases that give rise to segmentations in practice, this may not be a valid approach, and either an exact match (to be expected when the segment was derived from the image on which it is to be rendered), or full 3D resampling (to be expected when a segment is an entire 3D volume to be rendered on a different 3D volume than that from which it was derived) is more appropriate.

Calculations.

Distance Along Normal.

Treat the X, Y and Z values of the row and column unit vectors ("direction cosines") of Image Orientation (Patient) as (row_dircos_x, row_dircos_y, row_dircos_z) and (col_dircos_x, col_dircos_y, col_dircos_z) respectively, and treat the X, Y and Z TLHC values of Image Position (Patient) as a vector from the origin (tlhc_x, tlhc_y, tlhc_z).

The normal to the plane orientation is the cross product:

nrm_dircos_x = row_dircos_y * col_dircos_z - row_dircos_z * col_dircos_y

nrm_dircos_y = row_dircos_z * col_dircos_x - row_dircos_x * col_dircos_z

nrm_dircos_z = row_dircos_x * col_dircos_y - row_dircos_y * col_dircos_x

The distance along the normal from a plane passing through the origin is the dot product:

dist = nrm_dircos_x * tlhc_x + nrm_dircos_y * tlhc_y + nrm_dircos_z * tlhc_z

2D Image-Relative Location from 3D Patient-Relative Location Given Slice.

Given that we have already determined the target (image) slice, and know its geometry (TLHC and orientation and pixel spacing), we have two unknown values (row and column offset) and three known values (X, Y and Z location (3D segmentation TLHC) values).

This redundancy allows an approach of first finding the "major" (largest absolute value) X, Y or Z components of the row and column vectors of the target slice using its Image Orientation (Patient) values, and ignoring the contribution of the other (non-major) components, and ignoring the contribution of the normal (since we already know the slice).

Assign X an index of 0, Y an index of 1 and Z an index of 3, and use these to index the 3D location being looked up, the 3D image TLHC and row and column orientation vectors stored in arrays of length 3. Then define scalars R and C, that are the indices of the major (largest absolute value) row and column orientation vectors of the image.

Then, we can compute first compute the distances (in mm) along the row and column as follows (solution of linear equations by substitution):

distance_along_row = ((location[R] - tlhc[R])*row[C]/row[R] - location[C] + tlhc[C])/(column[R]/row[R]*row[C] - column[C])

distance_along_column = (location[R] - tlhc[R] - distance_along_row*column[R]) / row[R]

Then, we can compute the offset in pixels along the row and column of the target image by using the Pixel Spacing information (which contains two values, first the spacing between adjacent rows, second the spacing between adjacent columns, since pixels can theoretically be non-square).

row = distance_along_row / spacing_between_adjacent_columns

column = distance_along_column / spacing_between_adjacent_rows

Since the TLHC values in Image Position (Patient) are defined to be that of the center of the voxel, the row and column values thus derived will be zero if the location is that of the center of the voxel, -0.5 if it the TLHC of the TLHC voxel, 1.0 if it is the BRHC of the voxel (and the TLHC of the next voxel down and across), etc. This sub-pixel resolution may need to be accounted for if either an exact match to integer pixel indices is required (e.g., floor(offset+0.5)), or if the rendering pipeline addresses pixels with a different offset (e.g., (0,0) rather than (-0.5,-0.5) as the TLHC of the TLHC pixel).